**Buffer:**

Temporary small memory location used by node to perform operations on file system

**NodeJS Architecture/Event loop**: it contains Call Stack, Node API and Callback Queue

**Call Stack:** register every function and calls it.

Main() is default function inside this call stack.

**Node API:** every function inherited from c,c++ goes inside Node API like setTimeout().

Node API is an API(Application Programming Interface) because API’s work is to integrate c,c++ and JS in Node. It interacts with c,c++ libraries.

**Callback Queue:** it is a queue of node api function present in Node API. When main() gets destroyed , then callback queue pushes it’s function inside call stack. Then it executes.

**ExpressJS:**

Just like java has hibernate spring as frameworks, node has expressjs as a framework in it.

Used to reduce number of lines of code and is easy way.Easy API creation, easy access to routes, get-post-put methods all we get easily, easy creation of webpages, css integration all with less codes. It manages server and routes.

It is an API Gateway

Cmd to install:

>npm install express

Make it executable by calling express() after importing express module.like

Const express = require(‘express’);

Const app = express();

App.get(‘/about’,callbackfunc(request,response)=>{

Response.send(‘something’);

})-> provides routes

Express works with listening to port so

App.listen(5000);

Make sure to restart node when changes happen otherwise path will not get. Use nodemon for solving this problem.

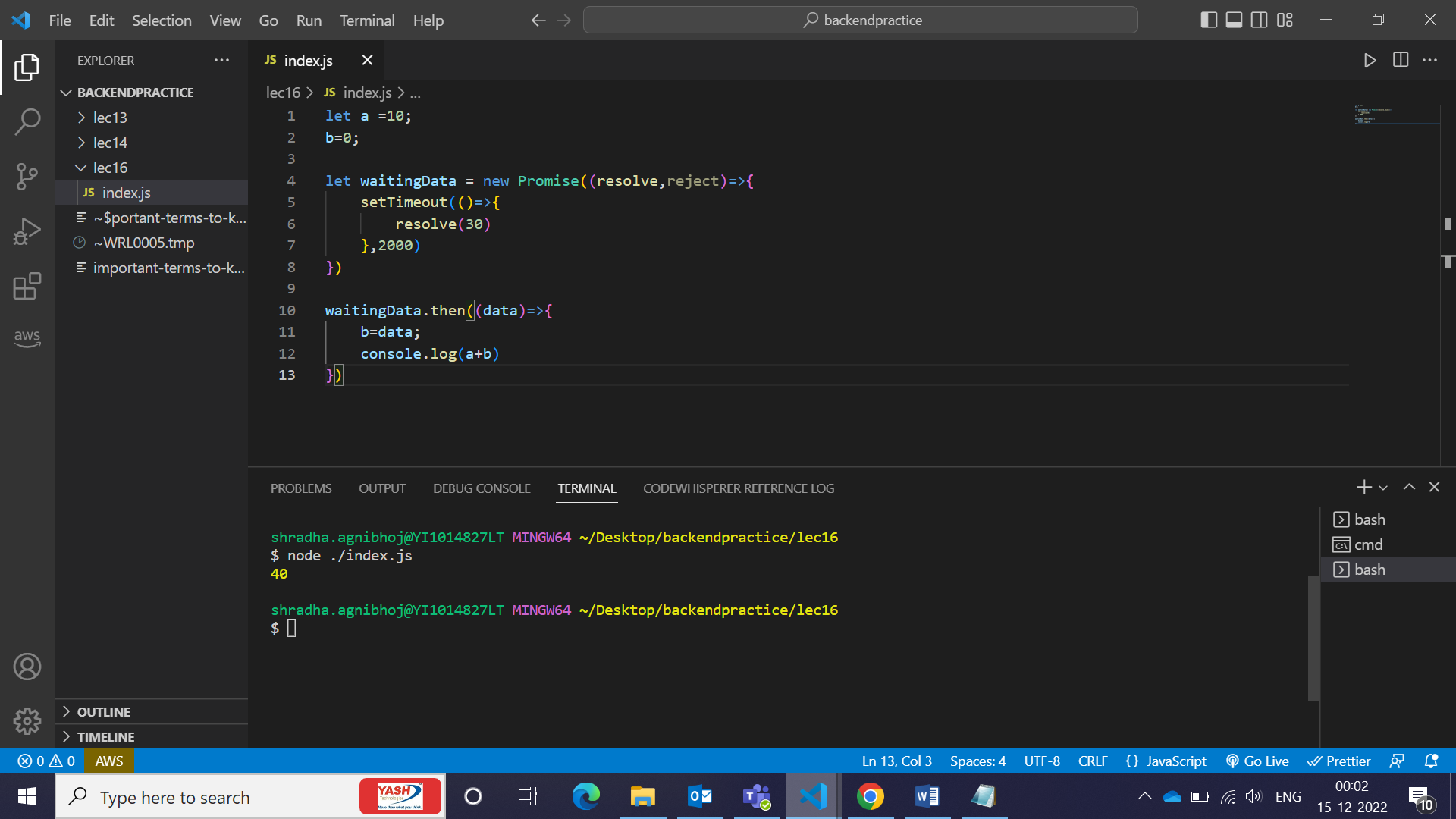
**Asynchronous and synchronous programming language:**

Synchronous-> a component or functionality queued below another component/functionality will work only when the first one completed and gives its response. Till that time the second one needs to wait for first one. This is time consuming and works with different threads like in java. Example: java, php

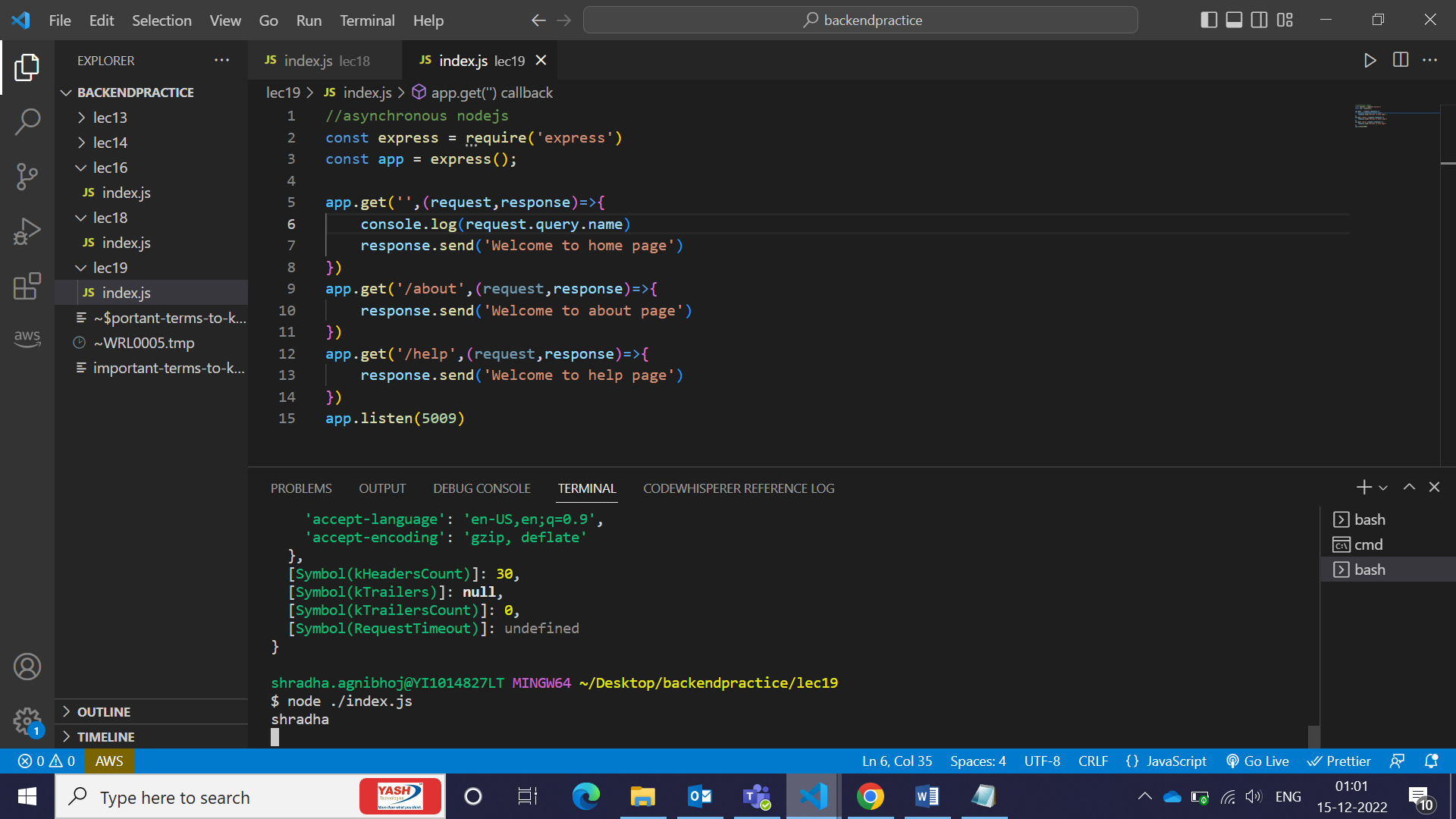
Asynchronous -> all components/functionality runs simultaneously inside one worker thread. As response comes for any of the components/functionality, it will return its response. So, it is fast.

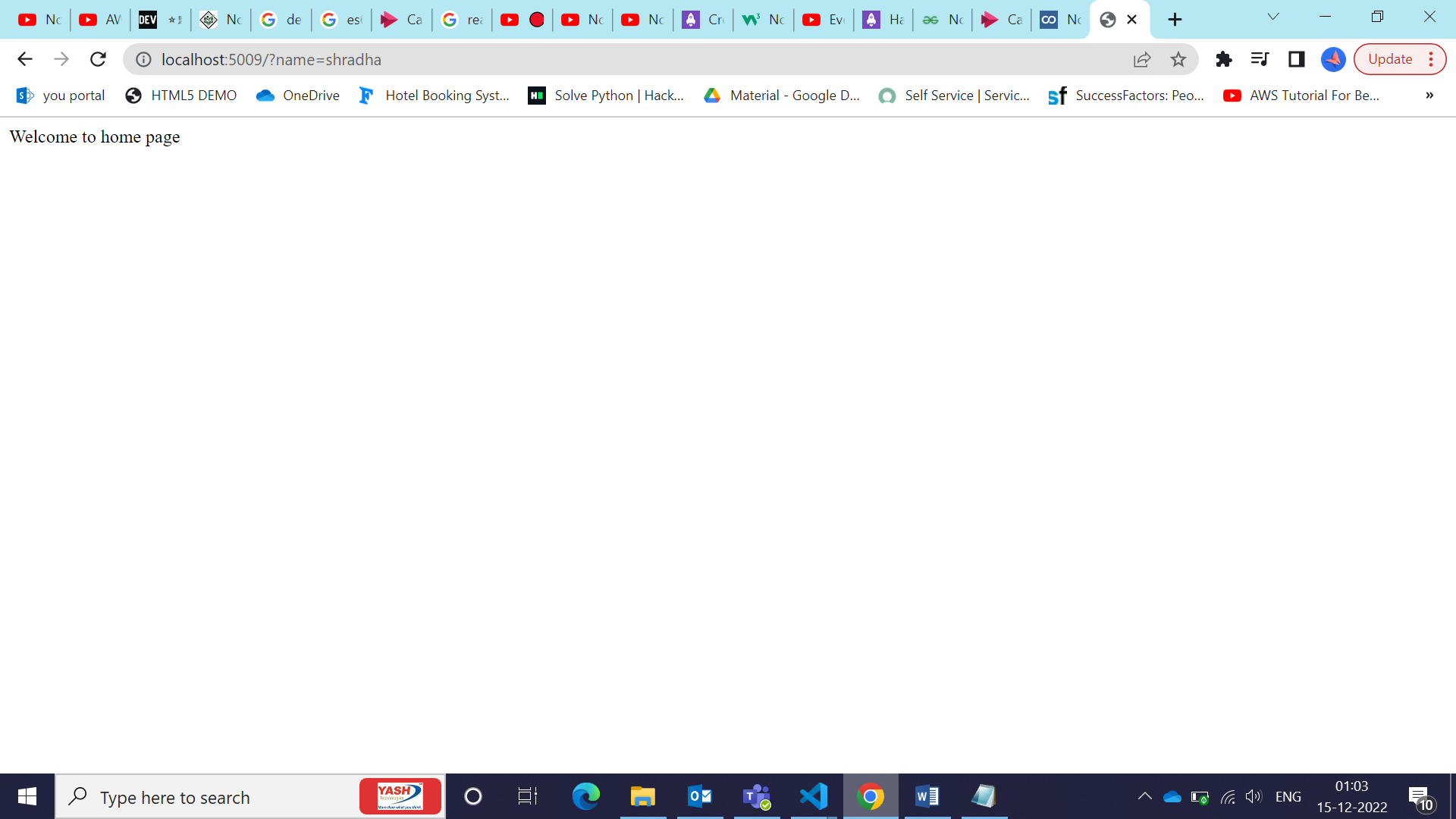
Example: JavaScript

Asynchronous has drawback that it doesn’t show what value we wanted after some operation due to asynchronicity (it moves to next line of code before completing with before line).

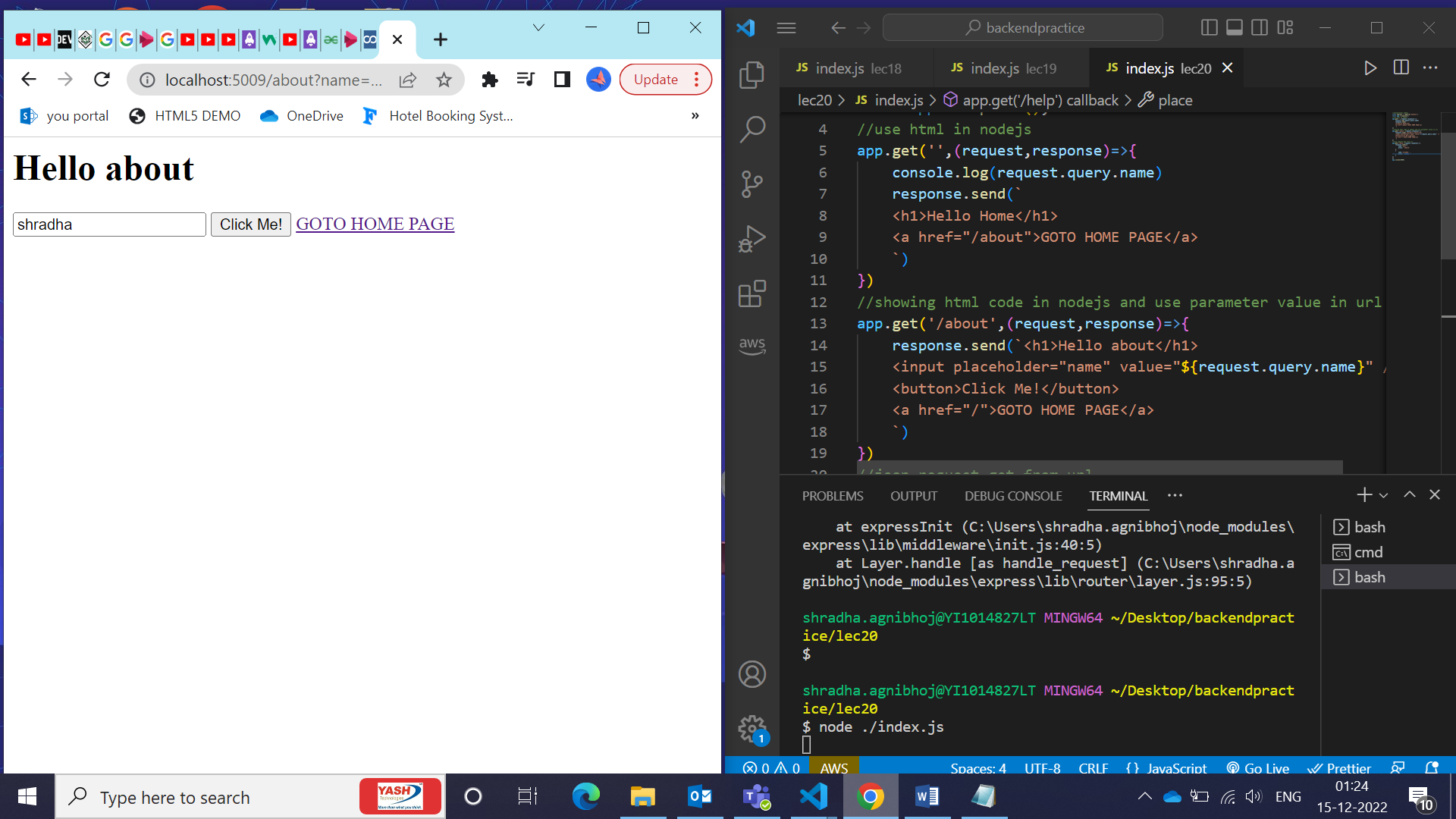


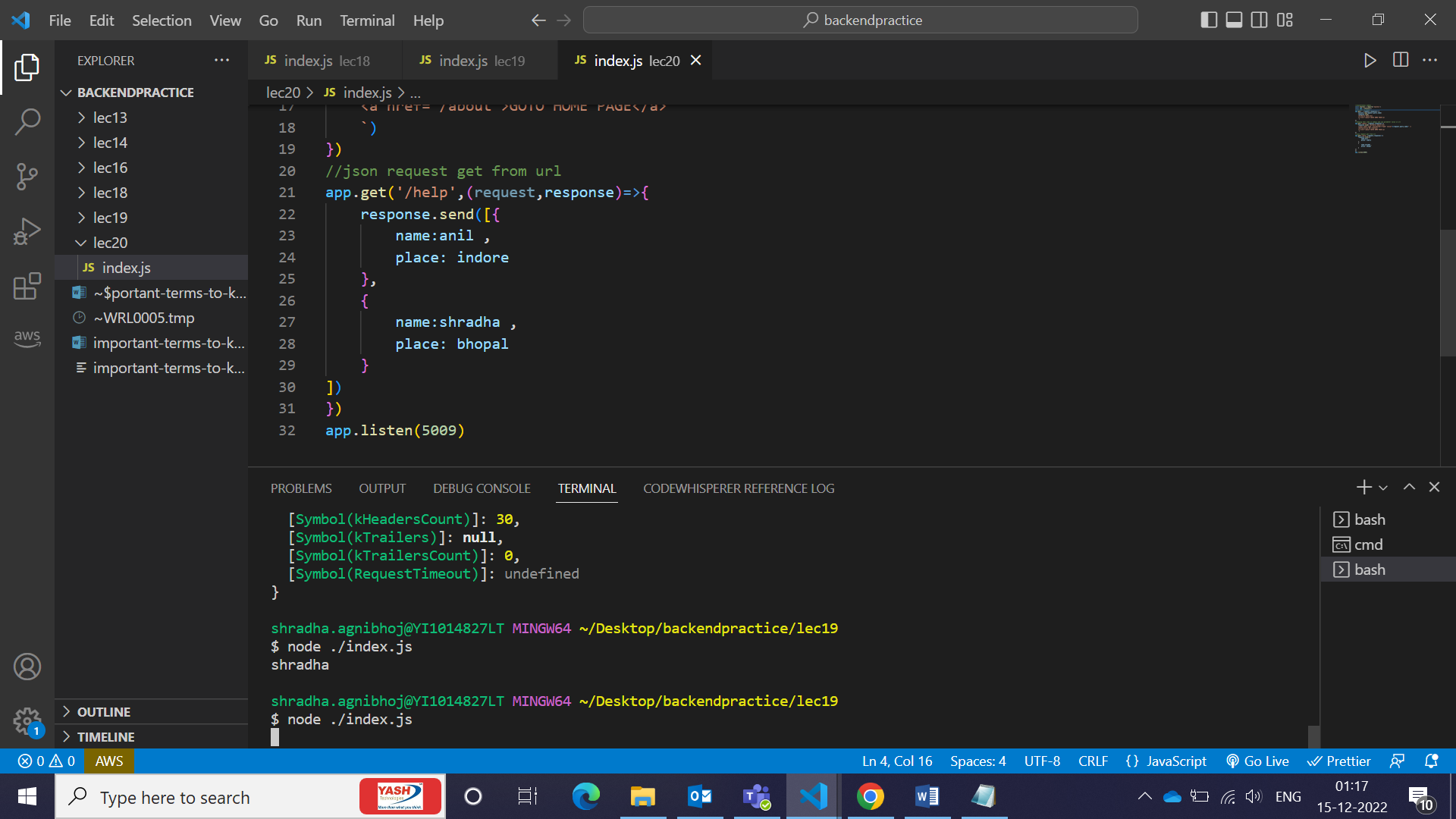
Using routing in nodejs via express:



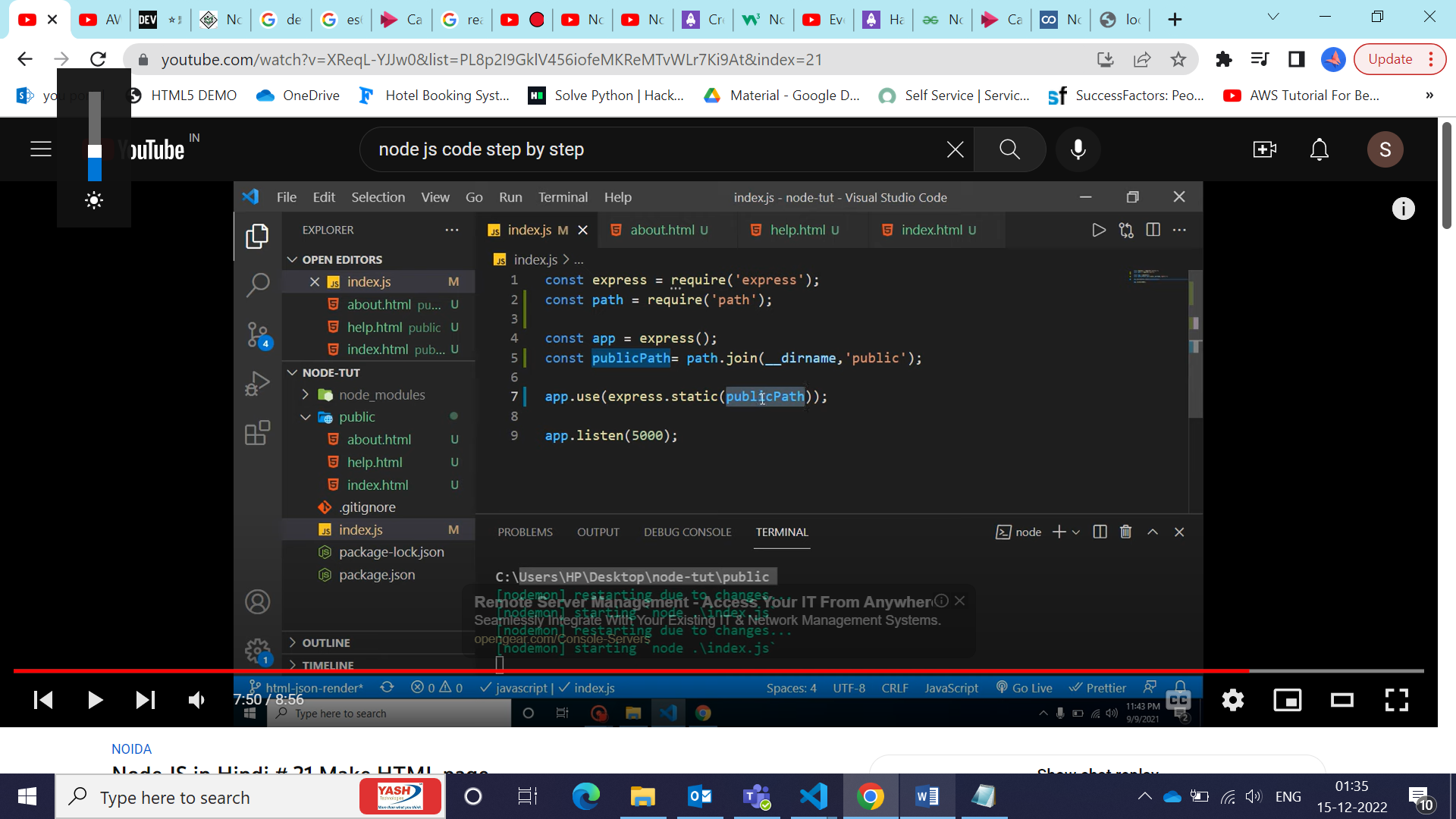


Lec 20 -> render/display html, links, json in nodejs in frontend



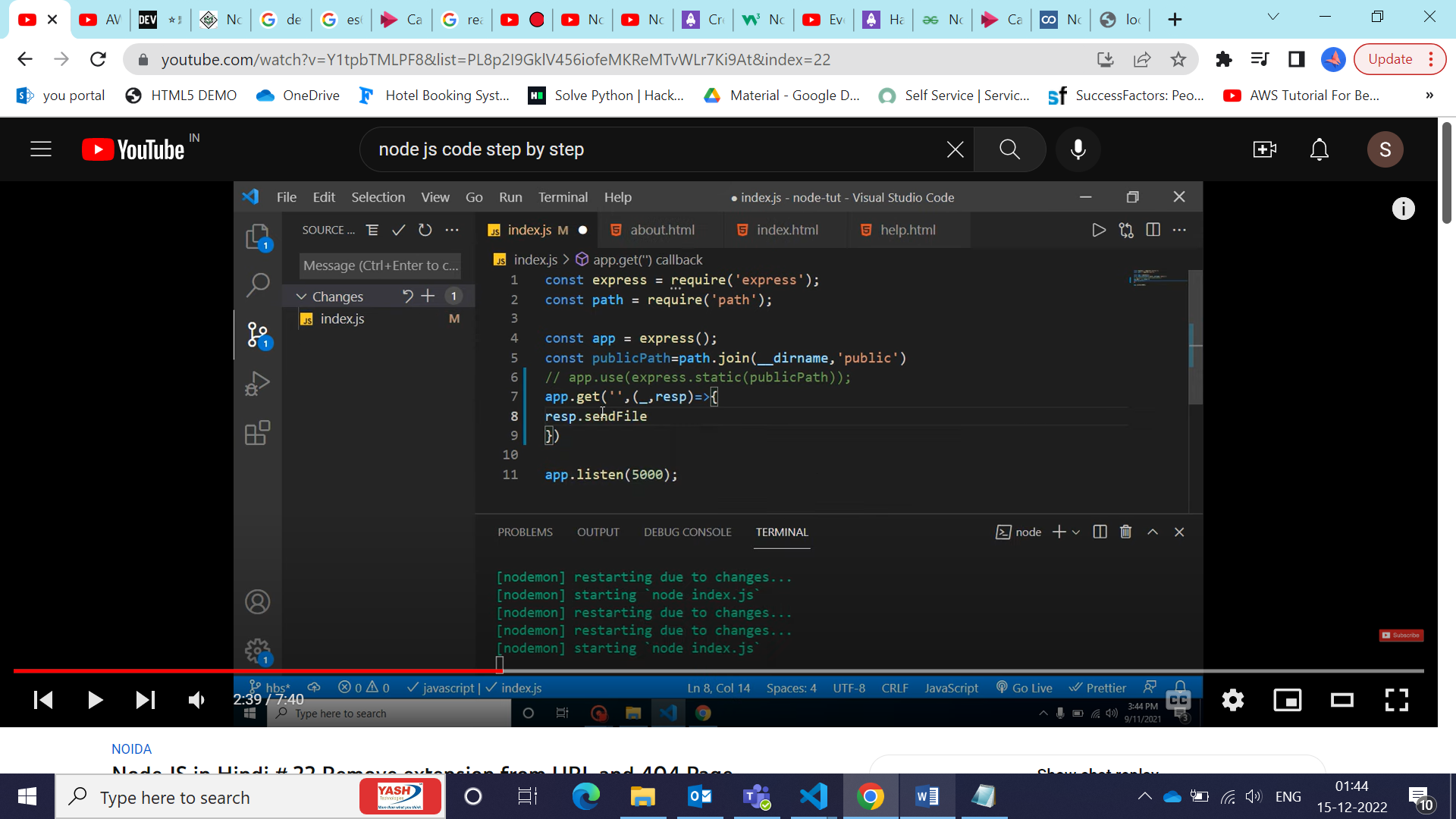


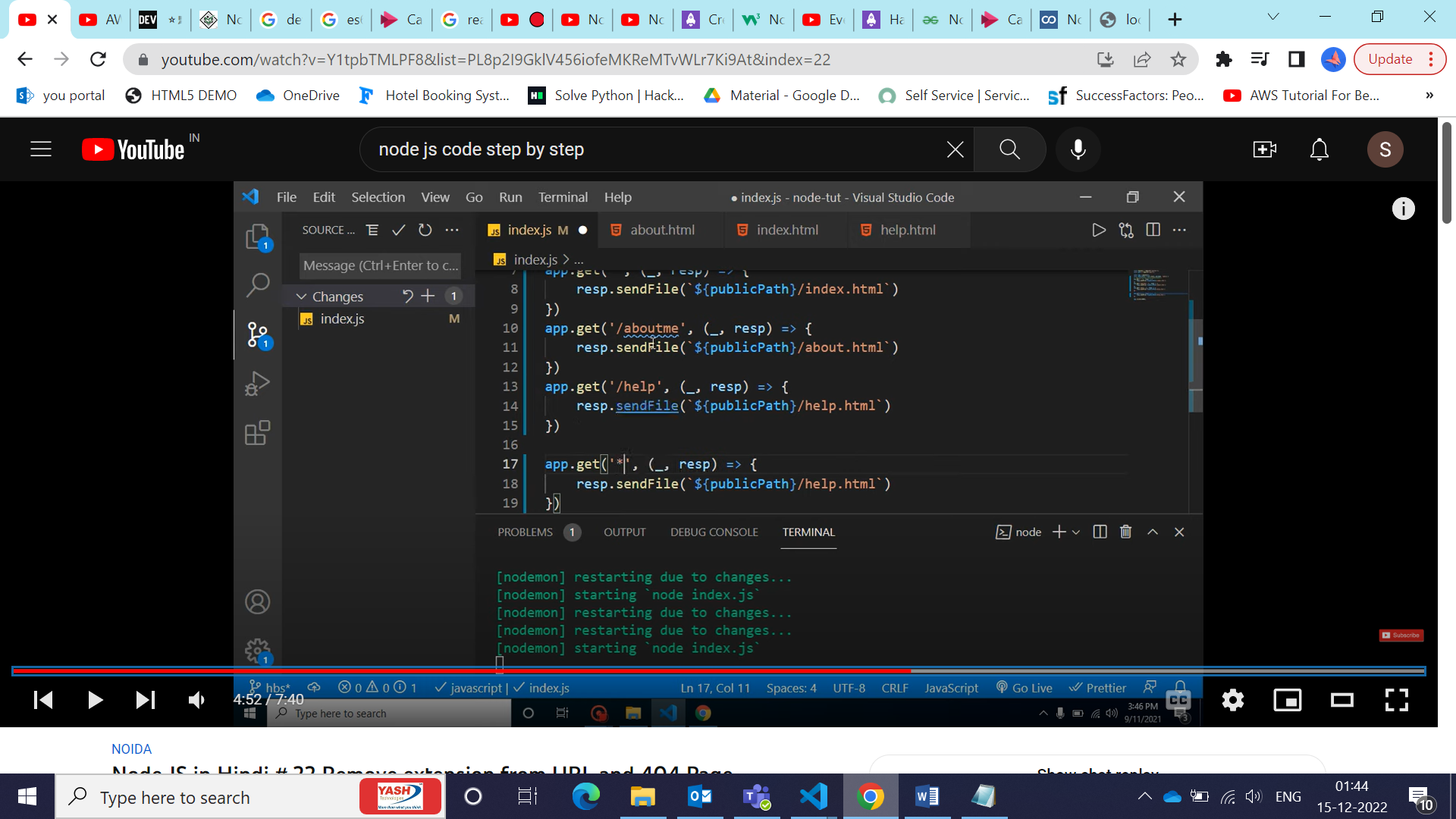
Path module lets us know where we currently are, lets us access path, filename using \_\_dirname and join(). It lets you use all paths in a particular folder,outside,inside, etc.



How express.static() is used?

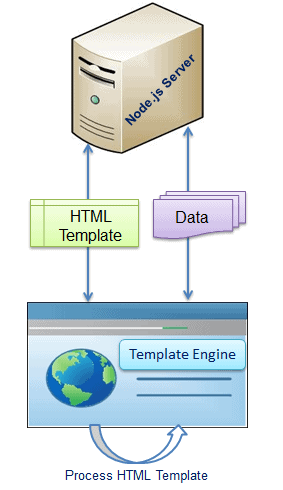
It loads static content/pages of file





**Template Engines in NodeJS:**

Template engine helps us to create an HTML template with minimal code. Also, it can inject data into HTML template at client side and produce the final HTML.



As per the above figure, client-side browser loads HTML template, JSON/XML data and template engine library from the server. Template engine produces the final HTML using template and data in client's browser. However, some HTML templates process data and generate final HTML page at server side also.

There are many template engines available for Node.js. Each template engine uses a different language to define HTML template and inject data into it.

Lec 21->

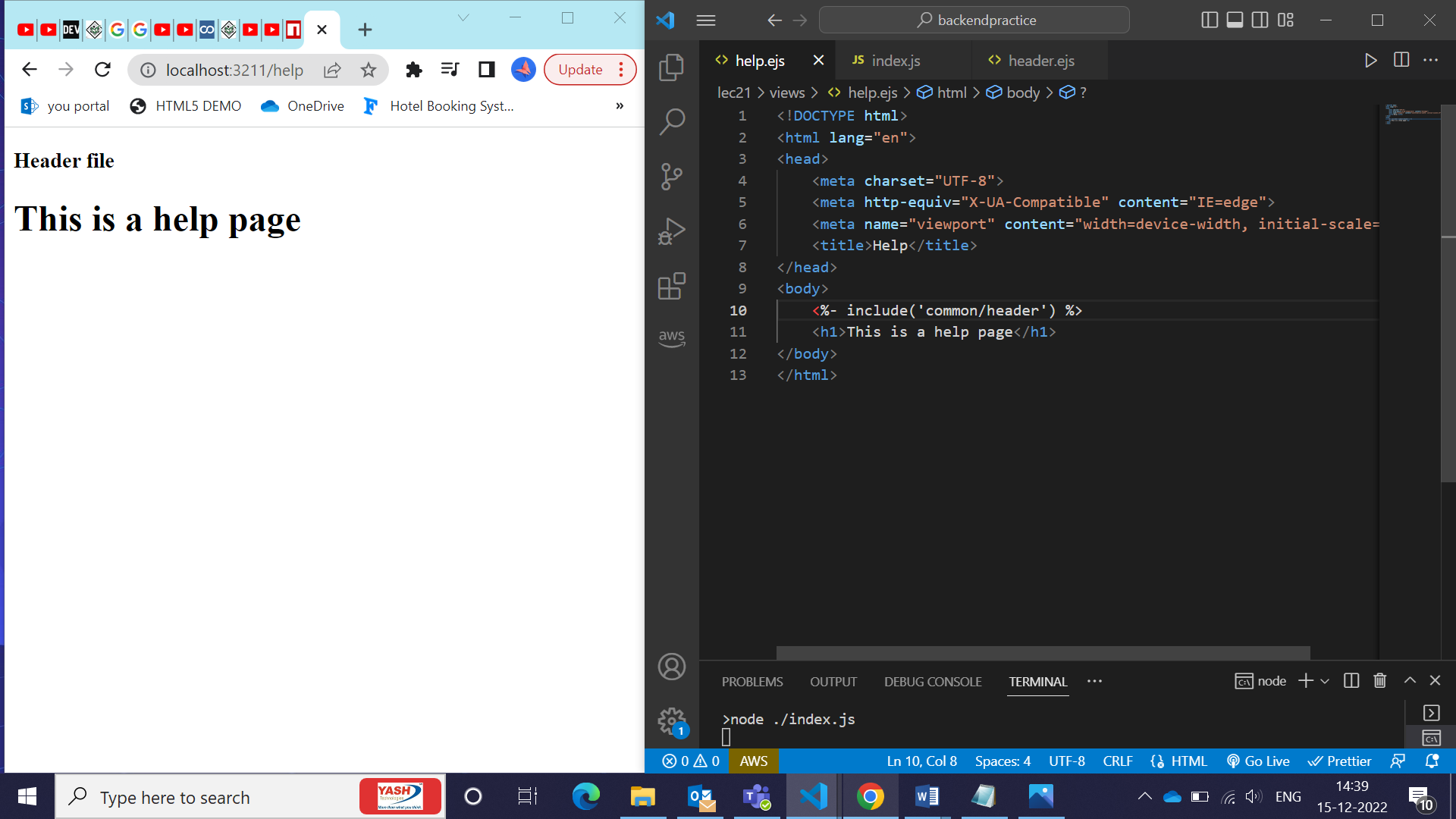
Template Engine helps us make dynamic pages. It can be installed using npm install and it is also a npm package.

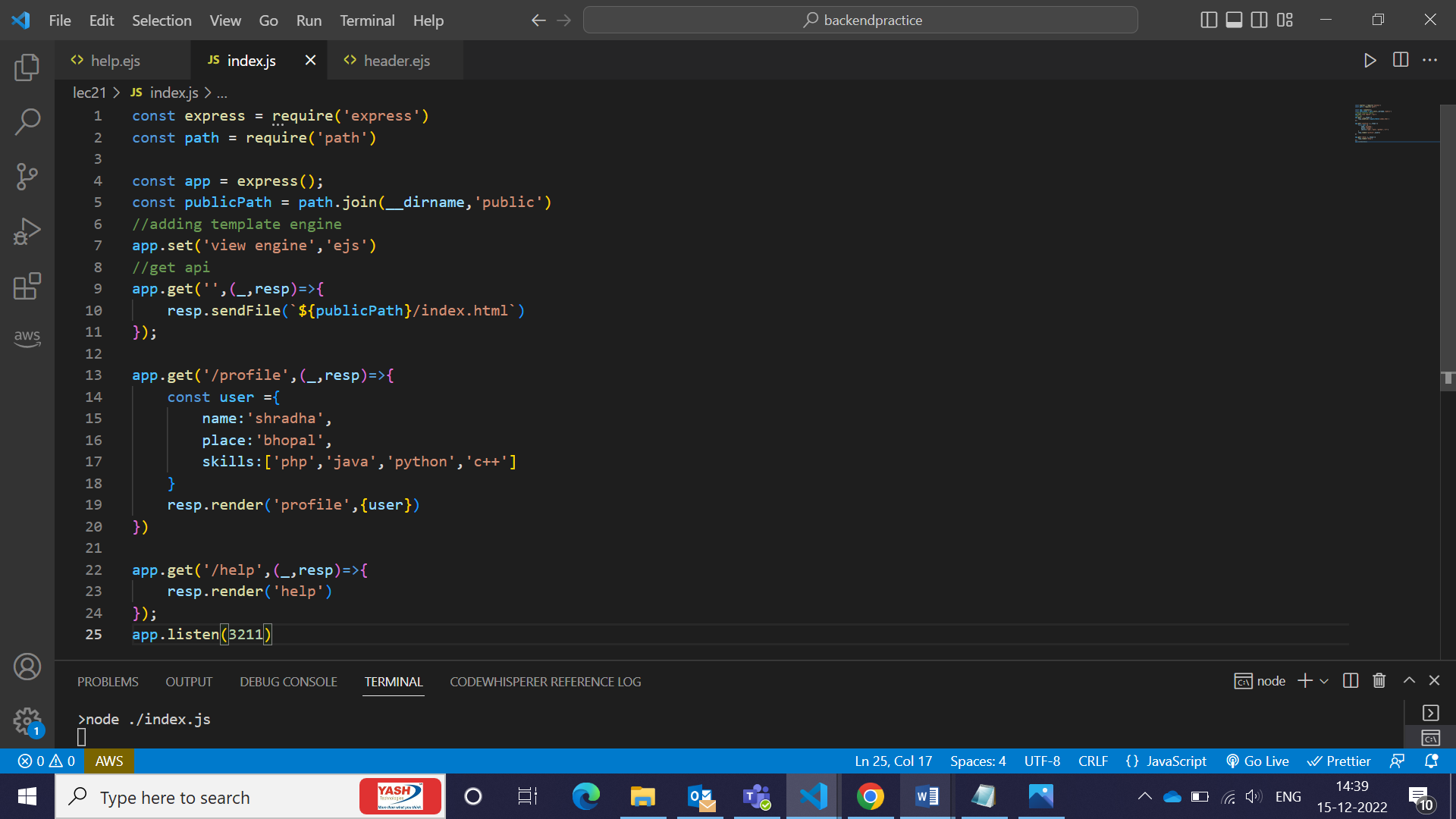
App.set(‘view engine’,’ejs’) -> mentions ejs(embedded javascript template) template engine is used. By default, views folder contains all dynamic codes. So, make views named folder only.

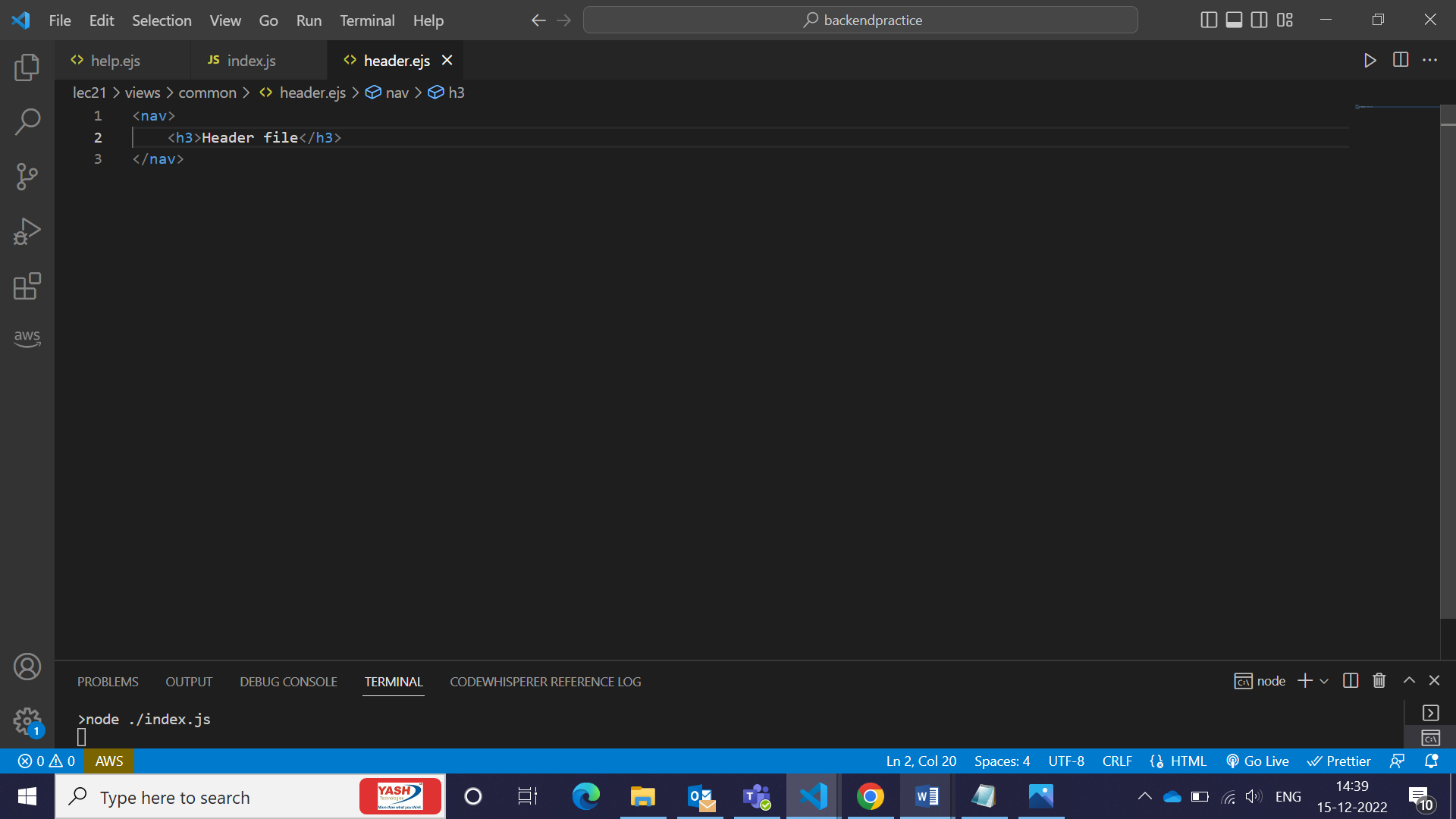
Filename filename.ejs written in html and data coming from connected with db.

Write data from js file in ejs file in <%= %>

Resp.render(‘profile’,{user})







**Middleware in nodejs:**

Middlewares are functions which are used with routing in nodejs. They work only with routes. Using it, we can access and modify request and response objects like for authentication of user that it I logged in or not, check age, check token, specific country we need to block our site, etc.

Same code can be applied in routes but everytime we will need to add in route so we have another function of middleware which we can make once and use multiple times.

Application-level Middlewares:

Middlewares which gets applied to all of the routes present in the project. Cannot be used in group of middlewares.

Globally applicable

Route-level middleware:

Middlewares that can be used on all middlewares or on specific middlewares or for specific group of middlewares

Can apply however you want

Queries:

1. Is there any other framework in NodeJS which is used on project level or is it just express used by major percentage of companies?
2. How async-await is better than promises?